# *The CAFS system today and tomorrow*

# ICL TECHNICAL JOURNAL

Volume 4 Issue 4
November 1985

# Foreword

On the 21st April 1985 it was announced that the Queen's Award for Technological Achievement had been presented to ICL in recognition of the successful innovation represented by the development of CAFS-ISP.

The problem of accessing large amounts of data in a structured manner, so as to present usable information within a reasonable time, began to be recognised as early as 1962. The CAFS system is a combined hardware and software solution to this problem. Additionally, it is especially well positioned because throughout the development there has been a clear understanding that any technological breakthroughs achieved in solving the problem must fit into the real world of existing customer data and equipment.

The CAFS-ISP product clearly represents a breakthrough worthy of the Queen's Award, and as such is one of the unique technological advances in the data-processing industry today. As a vehicle it is just now being recognised by other organisations trying to provide comparable data-access capabilities.

This issue of the *ICL Technical Journal* reviews the history of the development of CAFS from the first recognition and formalisation of the problem through to the evolution of the physical solution, and also the history of the supporting software. In addition it illustrates the use of the product in the real world, showing how this has extended from the original closely defined problem of data access to include the processing of text and other forms of information by real users to solve real problems.

*Asa W. Lanum*
Director and General Manager, ICL Applied Systems

# The CAFS system today and tomorrow

**G.McC. Haworth**
ICL Management Support Business Centre, Reading, Berkshire

**Abstract**

The CAFS search engine is a real machine in a virtual machine world; it is the hardware component of ICL's CAFS system. The paper is an introduction and prelude to the set of papers in this volume on CAFS applications. It defines the CAFS system and its context together with the function of its hardware and software components. It examines CAFS' role in the broad context of application development and information systems; it highlights some techniques and applications which exploit the CAFS system. Finally, it concludes with some suggestions for possible further developments.

'Search out thy wit for secret policies
And we will make thee famous through the world'
Henry VI, 1:3

## 1 Foundations

The late 1960s saw ICL's leading architects designing the new range of computer systems that was to become the 2900 Series. Specifically, those responsible for the operating system were required to:

- provide a cost-effective environment for developing and running applications
- adopt an architecture to fulfil changing requirements and exploit new technological opportunities.

From the start, the designers acknowledged the rapid and accelerating process of innovation which characterises the IT industry. They recognised that the common computing processes would migrate from application software via system software and microcode to special-purpose hardware. The result based on the twin concepts of the virtual machine and the procedure call was the virtual machine environment, VME.

File searching is clearly a common process and was identified as such in 1969. Maintained file-projection indexes and data infrastructure are not always appropriate or cost-effective for data-retrieval. Research in this area produced the CAFS.800 search engine and development led to the CAFS-ISP

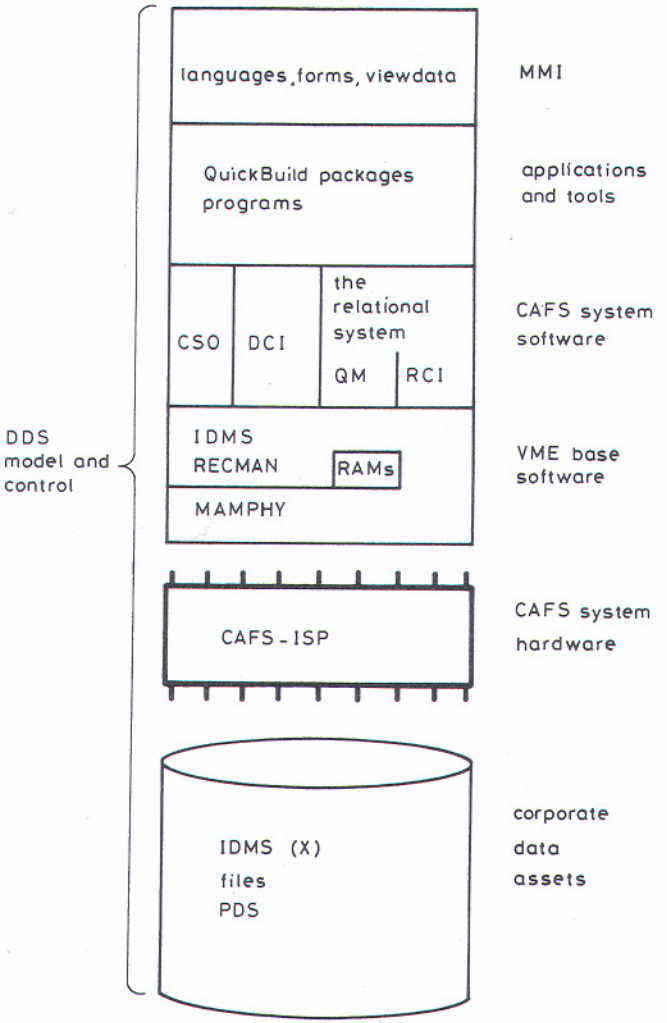


Fig. 1 The CAFS system in context

product[1]. The integration of CAFS-ISP into the VME virtual machine environment is evidence of the robust architectural foundations laid down at the outset.

## 2 The CAFS system

Initial success with CAFS.800[2-5] demonstrated the effectiveness of the hardware approach to file searching. It achieved a quantum leap in performance, searching at 'disc-speeds' and transferring a major part of the normal mainframe load from the central processor to the peripheral CAFS engines.

ICL set out to incorporate CAFS as a standard subsystem within VME. An engine specification similar to that of CAFS.800 was combined with synergy requirements as follows:

- retrieve records satisfying criteria expressed as a combination of Boolean logic and threshold functions
- calculate during the search such derived data as are commonly required
- return a 'projected' set of hit records
- search existing data, whether stored in files or databases
- co-exist with current hardware and current workloads
- operate below existing or industry-standard interfaces

The programme resulted in what is now called the CAFS system. This is defined here and currently comprises five elements, one hardware and four software, as follows:

- the CAFS-ISP hardware search engine, here abbreviated to 'CAFS'
- the VME CAFS search option, CSO
- the direct CAFS interface, DCI
- the relational system:
  Querymaster for *ad hoc* and production data queries
  the relational CAFS interface, RCI, an extension of Cobol

Fig. 1 shows the CAFS system in its context of target data, VME support and application software.

For the sake of brevity, it is convenient to use rather a large number of abbreviations in describing the system and its functions. These are explained when first introduced and listed in the glossary at the end of the paper.

### *2.1 CAFS engine*

This is attached to a disc control module (DCM), through which data not being searched by CAFS passes directly as normally. The engine consists of five main components, as shown in Fig. 2:

- the logical format unit (LFU)
- the key channels (KCs)

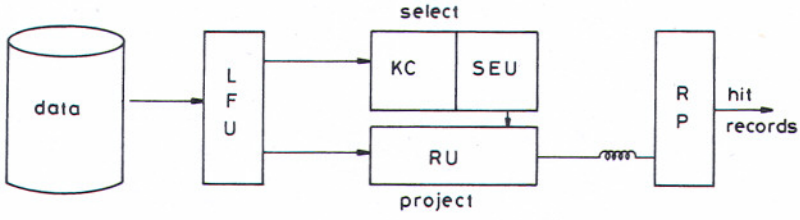


Fig. 2 The CAFS engine

- the search evaluation unit (SEU)
- the retrieval unit (RU)
- the retrieval processor (RP)

*2.1.1 Logical format unit:* The LFU provides input to the other components, advising them where to find relevant fields in the records. It has been given information about the data-file's logical block and record formats. It examines the incoming data stream, identifies starts and ends of records and fields and in some cases examines the content of the record.


Fig. 3 The self-identifying format for data

A record that can be searched by CAFS has a fixed-length part followed perhaps by a variable-length part. The latter may be a string, an array of fixed-length fields or data in self-identifying format (SIF). The unit picks out the appropriate types of data stored in this last format, illustrated above.

*2.1.2 Key channels:* The search criterion, i.e. the set of conditions that a record must satisfy to qualify as a 'hit', is of the general form

logical condition (LC) & {interfield comparisons} (IFC)

for example:

(LC) age < 30 & 2 from [experience = insurance, banking, audit] &
(IFC) achievement > target

LC combines Boolean and quorum terms using the Boolean operators 'AND', 'OR' and 'NOT'; the terms are evaluated from the results of component atomic conditions of the form:

FIELD masked by MASK is in RELATION to LITERAL
where the relationship is $=$, $\neq$, $>$, $<$, $\geq$ or $\leq$

The example above features one Boolean and one quorum term. The Boolean term has one atomic condition; the quorum term has three.

Two features add to the CAFS functionality. First, CAFS can detect whether records satisfy nominated subconditions within LC. Secondly, CAFS can mask a field to ignore unknown or irrelevant parts of the field; this endows it with a powerful fuzzy-matching capability.

In evaluating the complete criterion, the key channels and search evaluation unit deal with LC and the back-end retrieval processor deals with IFC.

A battery of 16 key channels performs the first part of evaluating LC; not all will be needed in every case. Each channel examines one atomic condition and signals in parallel with the others and via three bit-stores whether their masked field contains value(s) less than, equal to or greater than their literal.

None of the bit-stores will be set if the field does not exist, one will be set for a single-valued item and any number will be set according to the content of a multivalued field. For example, the text item 'QUICK BROWN FOX' in SIF format when compared with FOX will use the VME EBCDIC collating sequence to signal BROWN < FOX, FOX = FOX and QUICK > FOX.

The software surrounding CAFS uses some KCs; VME for example uses one. The LC can therefore involve a maximum of 12–15 atomic conditions.

*2.1.3 Search evaluation unit:* The SEU declares whether the record satisfies LC as well as the nominated subconditions within LC. It has a battery of 16 search evaluation processors (SEPs), programmed to operate in concert to indicate in a 'task word' whether the condition LC and/or the nominated subconditions are true or false. They are assisted in this task by two further subunits as illustrated in Fig. 4. The quorum processor (QP) evaluates all, possibly weighted, quorum expressions. The select processor (SP) broadcasts the QP results and interim SEP results to the SEPs.

Finally, the SEU increments counts of records satisfying the LC condition and the nominated subconditions by examining the task word.

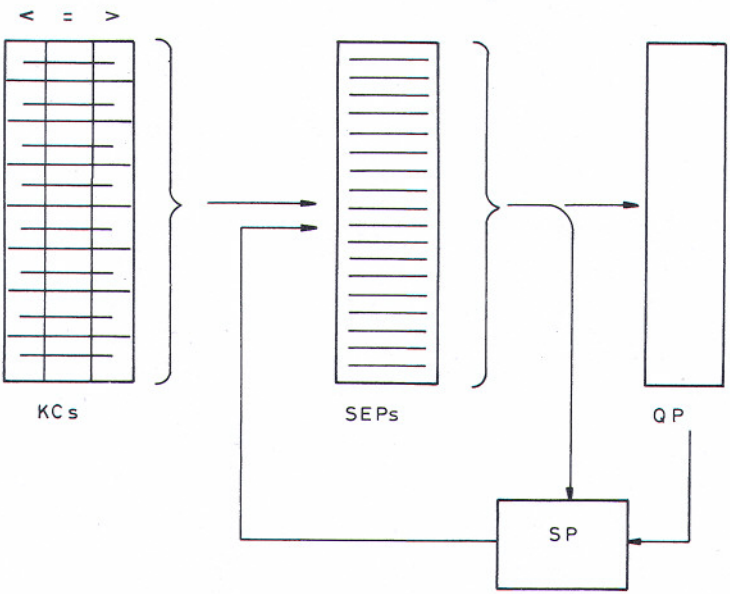


Fig. 4 The selection evaluation unit

*2.1.4 Retrieval unit.* This unit is told by the LFU which bytes of the record to retrieve as if that record had satisfied the LC. It stores retrieved records in the retrieval processor's (RP's) 32 kbyte store for analysis. The record is subsequently prefixed by an identifier, length and task word if it satisfies LC and discarded if it does not.

The RU also notes the end of a logical block of data in the RP store for checkpointing purposes.

*2.1.5 Retrieval processor:* Up to this stage, the CAFS engine works as a strictly synchronous pipeline processor. The RP's activity, however, is store-buffered as described above. This allows the front end of the CAFS engine in an extreme case to work thousands of records ahead of an RP examining a physical cluster of records satisfying LC.

The RP has two roles. One is to calculate data derived from the file search as a whole; the second is to be the final arbiter about passing data back to the host mainframe. These are separable functions; the RP, after returning a nominated number of records to the mainframe, can cease to retrieve but continue to analyse the whole file.

The RP may calculate a number of functions in a user-determined sequence. It evaluates the set of interfield comparisons constituting IFC. It computes the maxima, minima and totals of values in specified fields. At any point in this sequence, the record could be rejected as a candidate for passing back to the mainframe.

The data-delivery rate of current discs limits CAFS search speed except in high-hit-rate situations where the load on the retrieval processor can become a limiting factor. Record reduction is done by the RU in parallel with record selection but handling hit records and evaluating functions take time.

In summary, and in relational database terms, the key channel and search evaluation units jointly perform the SELECT operation and the retrieval unit performs the PROJECT operation.

It can be seen that the CAFS engine employs powerful, parallel and purpose-built hardware focused on a common task previously done by conventional von Neumann software.

The elapsed times of tasks unassisted by CAFS have been improved by factors typically of 10–100 when CAFS was introduced. At the same time, much if not almost all the work has been transferred from the central processor to the CAFS engine, as was intended; in one verified case, 99·94% of the mainframe load was removed.

Appendix 1 gives a more detailed model of the search and retrieval performance of the CAFS engine together with information showing how the

objective of coexistence has been met. The first ICLCUA report[6] includes the results of the first CAFS performance tests.

The following four sections describe the software interfaces to the CAFS engine which enable the prospective user to tap the search power of the hardware.

### *2.2 CAFS search option*

The objective of this software is to enable existing programs to use CAFS' search power without requiring any change to the code. Such programs will be ones which we do not wish to or cannot change. The former category includes operational programs which have been well run-in, low-priority 'one-offs' as well as the year-end undocumented antique of apocryphal importance and complexity. The latter category includes packages and applications generated by QuickBuild.

CSO is addressed to the classic batch suite of Cobol and Cobol-like programs which select records from a file using a known criterion and process the 'hit' records.

CSO allows selection on the basis of a Boolean-only criterion; quorum logic and interfield comparison cannot be used to eliminate records. Further, CSO cannot be used by the other three software interfaces to CAFS.

The CSO interface is a single system control language (SCL) command SET_CAFS_CRITERIA (STCC)[7] with parameters for defining the relevant record format(s) and the selection criterion. This causes selection intelligence and CAFS instructions to be interposed (Fig. 5) at record-access method (RAM) level between the target data file and the program.

For example, suppose a program ISSUEDEMAND processes a file MYMASTER containing records of several different types, each starting with a four-character field, TYPE. The program is only interested in records where:

TYPE = 4 and AMOUNT_OWING ⩾ £300 000.00

The following SCL boosts the performance of ISSUEDEMAND:

```
ASSIGN_FILE (MYMASTER, MAINFILE, LEVEL = C)
STCC (LNAME = MAINFILE,
      ITEMS = TYPE (1:4) & AMOUNT_OWING (9:9),
      CONDITION = "(TYPE EQ 4) and
      (AMOUNT_OWING GE 300000)")
ISSUEDEMAND
```

ISSUEDEMAND continues to evaluate returned records as normally

because it has not been changed in any way. It finds that each record is a hit and that the file appears to be only 0·001% or whatever of its previous size. The program completes its task a great deal sooner than before.

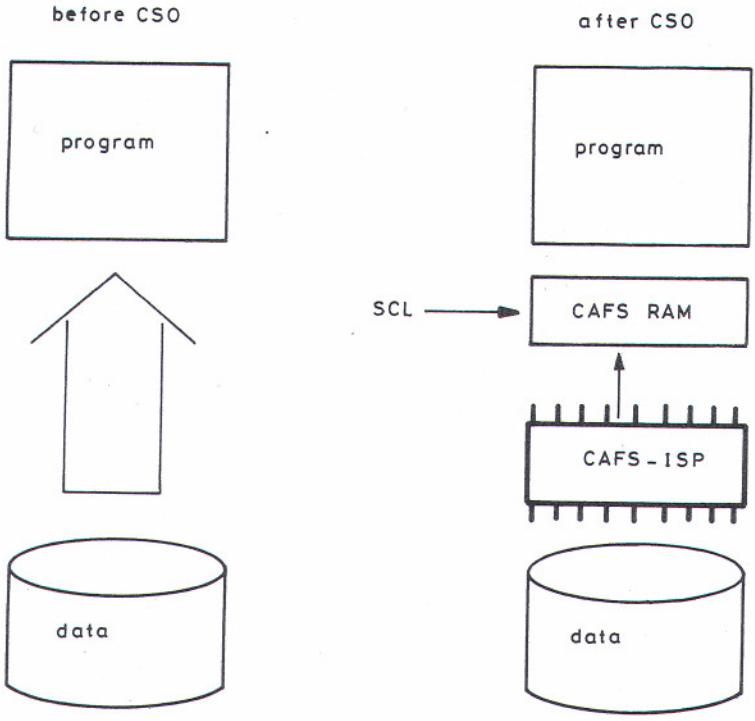


Fig. 5 Before and after CSO

The reduction in elapsed time and use of system resources is governed entirely by the hit rate on the file; the smaller the hit rate, the greater is the improvement. CSO users would normally expect to see a considerable improvement in performance, and with an interface consisting of one SCL command it is not difficult to try a number of experiments.

### *2.3 Direct CAFS interface*

The direct CAFS interface (DCI) is the only software interface with the objective of providing all the facilities of the CAFS engine to the user. It does so via the standard VME procedure call, which is almost universal and independent of the host language used. Character-string parameters in high-level syntax are passed from user-software to DCI at runtime and fully validated[8,9].

The other three software interfaces tailor CAFS' functionality to the needs of

their respective users for the sake of simplicity. For example, only DCI provides quorum evaluation, management of multiple types of SIF data, full subcondition counting and access to the full range of CAFS' RP facilities. DCI is therefore 'the closest approach to the original sound' and logistically the best way to gain hands-on experience of the CAFS engine itself.

DCI is the interface for high-level language programmers, for those not requiring the high-level relational CAFS interface and for the most sophisticated CAFS applications, possibly requiring the highest performance.

In addition to providing a procedural interface to CAFS, DCI provides facilities in the following categories:

- file management: open, read and close file
- TP transaction management: save, restore, end phase etc.
- text management: SIF to/from legible conversion, trailers
- DCI package runtime control: checks and diagnostics

The minimum sequence of DCI tasks, not using CAFS RP functions, will:

- open a file for CAFS searching
- define the target record's format and retrieval requirements
- define the search criterion LC, IFC being null
- read the next retrieved record
- close the file being CAFS searched.

Three criteria using three subconditions indicate the range of DCI's selection capability:

C1 = FEATURE EQ 'dglazing'
C2 = FEATURE EQ 'garage'
C3 = FEATURE EQ 'gardens'

- Houses below £50 000 in Finetown with two of the three features:
  PRICE LT 50000 and TOWN EQ 'FINETOWN' AND QUORUM THRESHOLD 1 Cl C2 C3
- Must have garage and double-glazing: anything near Crewe or a Cheshire house not in Shambletown:
  C1 AND C2 AND (POSTCODE EQ 'CW!' OR
  (TYPE EQ 1 AND COUNTY EQ 'Cheshire' AND TOWN NE 'SHAMBLETOWN'))
- How many properties have double-glazing or garage or garden?
  Define LC = C1 AND C2 AND C3 AND FALSE to reject all records in RU
  Request counts on the three subconditions C1, C2 and C3

DCI needs CAFS to access data, in contrast to the relational system which uses both CAFS and non-CAFS modes of data access. It addresses only

record manager (RECMAN) files but it is ICL's intention to extend it to address IDMS databases.

To date, DCI has been used from Application Master, Reportmaster, Querymaster, Cobol, Fortran77, Pascal, RPG2, SCL, Algol68 and from Filetab and Rapport.

### *2.4 Relational system*

In contrast to the two CAFS interfaces so far described, ICL's relational system was not designed especially for CAFS. Its objectives and methods were established[10] in the knowledge of parallel work on data models and CAFS.800 but before the appearance of the current CAFS engine. A common emphasis on data retrieval and the ability to search existing data provided the basis for the successful integration of the relational CAFS engine and the relational software. The results obtained should encourage ICL's customers to bring the power of CAFS to their existing and developing systems.

ICL's complete relational system is provided as three separate products; these are the personal database system PDS, the end-user query language Querymaster (QM) and the relational CAFS interface (RCI), a Cobol language extension.

PDS is classified here as part of the CAFS context rather than part of the CAFS system. PDS data is searchable by CAFS but the PDS software itself does not include CAFS-search capability. QM and RCI make automatic and transparent use of CAFS as appropriate.

The objectives of ICL's relational system were:

- to widen the community of data users
- to mesh with other data management products and preferred system development methodologies
- to retain the flexibility to exploit new techniques.

The aim of providing effective access to a wider community implies that the data should be presented to users in a simple and uniform way and that the special knowledge required of the user should be reduced to a minimum. Low-level details, for example of data storage, should be of interest only to the small minority responsible for providing a relational data-access platform to the majority. Keeping detailed information in place is essential in the division of labour necessary in any organisation. Here, to provide read-access to data, it is only the senior analyst/programmers and query-service providers who work below the relational level with more intimate details of data storage.

The relational system data model presented to the users includes a tabular

data structure without user-visible navigation links between tables. The tables presented to the end-user or the Cobol programmer are the results of using the relational operators SELECT, PROJECT and JOIN. This data model conforms with Ted Codd's restated definition of a relational database[11]. It also adopts a duality principle by allowing programming uses of the relational interface to be tested out interactively.

The data model is enriched by knowledge of the inter-relationships between the entities represented by the tables of data[12,13]. It is therefore not necessary for the Querymaster user to join tables explicitly unless there is some choice or ambiguity about the inter-relationship of one table with another. This results in shorter queries and less risk of user confusion. RCI users enjoy analogous benefits.

The relational products intended for use with shared data are designed to integrate with and work off the data dictionary system DDS. Data dictionary systems are increasingly being regarded as necessary control centres modelling both the using organisation and its systems development. DDS provides documentation on the accessibility of data by groups of users and avoids the need to duplicate information on the format or storage of data. In the case of CAFS, new dictionary definitions have been introduced to document the encoding of text data in CAFS SIF format and to control CAFS searching of IDMS areas.

The relational system addresses existing data in both IDMS and RECMAN formats; here, therefore, a high-level definition of a logical database has been coupled with a Codasyl (Conference on data systems languages) definition of a physical database and the resulting system enjoys the complementary benefits of the relational and Codasyl approaches.

It is part of the marketing mythology in the industry that the Codasyl and relational approaches to data management are in conflict; in fact they address different levels of database definition, as noted above and illustrated in Fig. 6, and fit well together, as this work has shown. The theory of update logistics and semantics at the relational level is incomplete; ICL has therefore retained the Codasyl standards for update and data-integrity enforcement.

The technological flexibility of the relational system is amply demonstrated by the integration of CAFS support and by subsequent developments such as the interface to graphics facilities on a workstation.

The design of the CAFS support for the relational system had to balance the great benefits offered by CAFS against the preservation of a simple and consistent view of data wherever these came into conflict. Every function offered by the system is supported whether CAFS is available or not in order to simplify the user interface. The system maps the large variety of Cobol data types to a smaller range of relational data types, regardless of the CAFS engine's ability to handle the physical data.

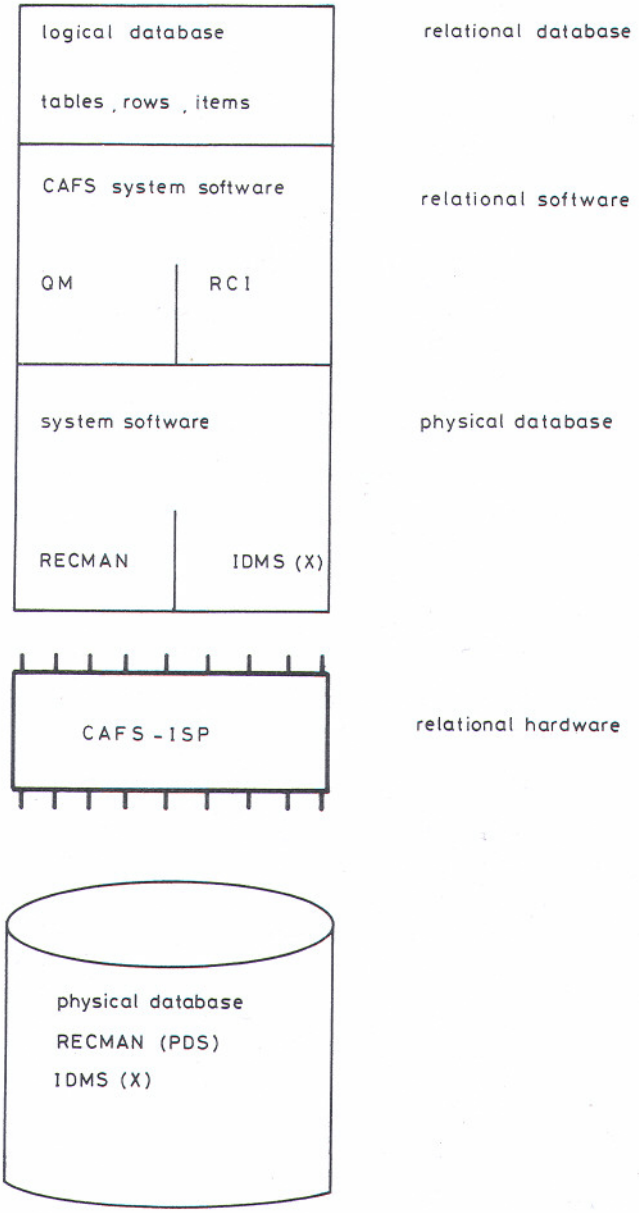


Fig. 6 Physical and logical databases

In practice, a major part of the CAFS functionality has been exploited in supporting the relational model. Comparisons are supported on all data types and the ability to handle the SIF format has been made available in its single most valuable manifestion – word-searching of free text with fuzzy

matching. The quorum capability of CAFS has not been made available directly as it does not fit easily with the implicit approach to the specification of joins. The CAFS retrieval processor is employed to count and make interfield comparisons. The combined use of (partial) primary key selection and CAFS searching is the most important performance optimisation introduced by the RECMAN CAFS software; it is faithfully exploited by the relational system.

In the IDMS context, the relational system and the IDMS database manager co-operate to perform the necessary selection and retrieval. The user's independence from the choice of RECMAN or IDMS data storage has been preserved. The introduction of CAFS support has not altered the form in which end users pose their enquiries except for the provision of word-searching and fuzzy matching.

The availability of CAFS influenced the relational system's rules for optimised data retrieval. The system looks for the best 'opening move', as most enquiries allow no choice of navigation path after selecting the first record type. This is a good approach where hit rates are small; the optimiser does not have volumetric information on which to base its choice, but users can give specific advice on the 'opening move' where this is appropriate.

The relational system on the whole uses designed-in data infrastructure such as record keys and IDMS sets and indexes. It supplements these data access techniques by using CAFS for serial or partly keyed scans whenever possible. Both access optimisation and CAFS exploitation are rule-based strategies and are defined in detail in the product manuals[14,15].

*2.4.1 Querymaster:* Querymaster provides a wide range of users with an online relational query service to shared data, typically an IDMS database with additional RECMAN files. The user conducts a dialogue with the product to select a query view; within that query view, the user can explore the availability of data and select and retrieve data by means of enquiries in a simple language. The product displays and prints tabular data, stores temporary results, sorts and provides summary information such as totals at required control breaks.

The user's task is much simplified since Querymaster selects the navigation path and resolves names to decide which record types are to be accessed, how they are to be joined and where CAFS is to be used. A query is presented as a simply structured command with data selection conditions following the keyword 'WHERE', e.g.:

```
LIST CUST-NAME, ORDER-NO, ORDER-DATE, QUANTITY,
  PRODUCT-DESC WHERE COUNTY STARTSWITH 'LANC'
  AND ORDER-DATE = 1.10.84 TO 31.10.84
```

The query view is created by the VME command CREATE_QUERY_VIEW

from a definition in a DDS data dictionary. When the shared data is fully described in DDS, the task of creating and documenting a new query view is reduced to the selection of data to be viewed and the tailoring of that view to the particular requirements of its users.

Querymaster supports the online nature of enquiry not only with CAFS but also with comprehensive 'help' facilities and parameterised macros for the significant proportion of repeatedly used 'production' queries.

The combination of Querymaster and CAFS, as illustrated by Corbin[16], enables end-users across a wide ability spectrum to express their data-retrieval requirements and satisfy their substantial latent demand for timely information.

*2.4.2 Relational CAFS interface:* RCI is a Cobol language extension, providing both a program interface to CAFS and a read-only relational interface to data. RCI looks like a serial-file handler to the programmer who is presented with relational views (qv SQL) of the data. Behind the relational interface, RCI is using the standard SELECT, PROJECT and JOIN operators to compose these relational views.

The principal objectives of RCI were to provide:

- a transparent Cobol interface to CAFS
- a read-only relational interface to IDMS/RECMAN data
- value-based privacy to add to Codasyl's item-based privacy
- additional program/data independence for simpler maintenance
- simplified programming and higher levels of productivity
- selection on and processing of text fields

The Cobol programmer manipulates the data views through four new verbs provided by the Cobol system:

| | |
|---|---|
| START | creates an instance of the view, fixing selection parameters |
| READ | delivers the next record of the view instance |
| SAVE | preserves the state of a view instance at the end of a TP phase |
| RESTORE | restores the view instance state at the start of next TP phase |

The data environment of an RCI-enhanced Cobol module includes a set of relational views known as an application view and defined in DDS. The latter is analogous to Querymaster's query view and can be tested using Querymaster.

Other features of RCI are common to Querymaster and have been covered above in the section on the relational system.

## 3 The CAFS context

Fig. 1 illustrates the wider system of which the CAFS subsystem is a part. The four elements of this are:

- the operating system: VME
- the physical data-management systems: PDS, RECMAN & IDMS
- the required man-machine interfaces (MMIs): languages, forms, Viewdata
- the system-development control mechanism: DDS

VME provides support for the CAFS system by managing resources at the record (RECMAN) and physical magnetic media (MAMPHY) levels. VME also provides job control, operator control and monitoring facilities related to CAFS.

An important synergy objective for the CAFS system was that it should address existing data on files and databases. Physical data on VME is held in personal databases (PDS), RECMAN files and IDMS/IDMSX Codasyl-standard databases.

PDS databases are in fact implemented in a published format over a set of index-sequential (ISAM) files. PDS users can deploy CAFS on PDS-held data by using DCI or, via a DDS-held retrodefinition of the files, the relational system.

The range of CAFS-searchable standard file types includes the most common ones including serial, ordered serial, index sequential and hash random but does not include files containing spanned records or nonembedded keys. The primary key of an ISAM file can be used to focus a CAFS search to a small range within the file. This facility implies that the primary key should be chosen not only to distinguish one record from another but also to provide the most useful physical record clustering within the file.

IDMS databases are made CAFS-searchable on an area-by-area basis. The relevant areas are reformatted in a single-pass process by resequencing the order of information within each page. CAFS' projection facility is used to convert physical IDMS records into subrecords as defined in the IDMS subschema; there is therefore a strong argument for tailoring subschemas to anticipated CAFS searches.

Computer systems are today being made available to a wider range of users than ever before. This has raised the relative importance of the man-machine interface in the considerations of system designers. CAFS facilities have been provided below the three key interfaces of conventional language, forms and viewdata.

Finally, it is commonly recognised today that the activities of system development need to be co-ordinated around a central model of the host

organisation and its computer systems. The data dictionary system (DDS)[17] continues to be the key ICL component for modelling and controlling all phases of such development, driving the use of all the products in the CAFS systems and adjacent to it; the preceding discussion of the relational system gives a good example of its role.

## 4 CAFS exploitation

At the moment, the number of CAFS engines ordered runs well into four figures. CAFS-capable systems can be found in all sectors of the market served by 2900 and Series 39 mainframe computers; no one sector of central or local government, public utilities, health, manufacturing, retail, insurance, education, police or defence dominates the others, nor does any single type of application dominate. There are simultaneous trends to extend the usefulness of existing systems, perform research and analysis online, create end-user services and enhance searching in operational systems – in line with the four areas of activity targetted by the four software interfaces CSO, DCI, QM and RCI. The escalation of this activity confirms the original premise that data searching is an unavoidable computing process of fundamental importance.

The application papers which follow in this issue examine specific systems which have been developed. The following notes cross-reference these applications, introduce others and highlight some useful techniques for CAFS exploitation. The second ICLCUA CAFS User Group report[18] is another useful source of application examples and techniques.

### *4.1 Using the CAFS search option*

CSO can be used in conjunction with Cobol programs and with the QuickBuild components Application Master (AM) and Reportmaster (RM) which open files in Cobol style.

It is not necessary that the selection condition employed in the Cobol program should be expressible in CSO terms. As CSO acts as a primary filter, it is only necessary that the CSO condition should be identical to or weaker than the program condition. If this is not so, use of CSO will not pass all hit records to the program and will implicitly change the role and the output of the program. Where the CSO condition is a weakened form, the program will probably reject some of the records it receives. An extreme example is where CSO is only used to retrieve records of the right type from a multirecord-type file.

### *4.2 Using the direct CAFS interface*

This section is confined to DCI-specific techniques for exploiting CAFS. It focuses on DCI's role with regard to research activity, weak typing, text, quorum searching, data profiling and software packages.

Burnard[19] and Walker[20] describe research activities in the arts and sciences which have been facilitated by DCI/CAFS. In both cases, investigations previously conducted in a batch offline stop-start mode are now being completed online. Researchers are now interacting with their data, continually testing and refining their hypotheses in a more creative environment. Clear evidence is available that CAFS is not just speeding up the logistics of research work but making it more penetrative and productive.

Research work is not of course an academic monopoly; in an increasingly competitive world, all organisations are seeking to use their resources more effectively. The ability to manage large volumes of semistructured data is a key asset in this context and is assisted by the next technique to be discussed.

The CAFS-searchable self-identifying data format (SIF) provides a latebinding mechanism whereby we can defer typing data too strongly at the data-modelling stage. For example, consider the hierarchy of object categories in Fig. 7.

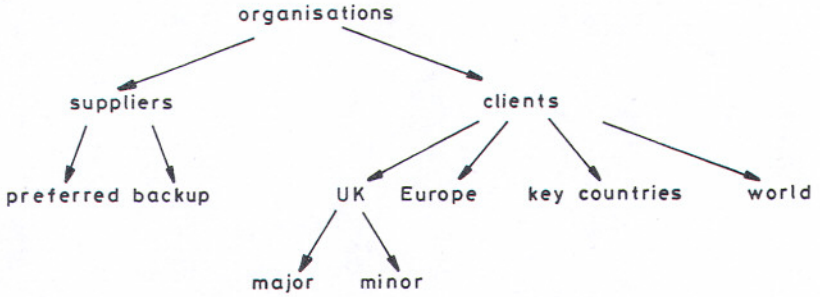


Fig. 7 A hierarchy of object categories

The categories vary from generic to specific as we move down the tree, a data model which preserves more meaning than the normal single-level entity model. Given another perspective in the model, the hierarchy becomes part of an object lattice. The data model supports queries which will vary from generic to specific; a mechanism which can type the data to suit the query in hand at runtime is clearly desirable.

SIF data, as the name implies, precedes each item of data with an identifier; data is no longer strongly typed by position. CAFS can mask the identifier just as it can mask the data values; it can ignore irrelevant or over-specific aspects of the categorisation. By a suitable binary choice of identifiers, specific categories can be converted into generic categories by CAFS masking.

Text management is one area in which varieties of 'data' will naturally occur. DCI's ability to search text is already being fully exploited and is described in detail elsewhere[18,19,21]

We examine now the uses of quorum searching. Whereas Boolean selection is

for exact data matching, quorum selection is for speculative searching. A defined but variable threshold allows records to qualify as sufficiently interesting on the basis of their proximity to some ideal target. Different factors can be weighted to reflect their relative importance. Boolean selection is appropriate when the data is determined by the selection criterion; quorum selection is better when the selection criterion is determined by the data.

For example, a personnel department will be seeking the best available candidate; a recruitment consultancy will respond with a shortlist of credible candidates; a detective is concentrating on the most likely suspects; a company naming a new product chooses a name which cannot be confused with those of its competitors' products; an inventor seeks out patents which are adjacent to the topic of his invention. Again, quorum selection is preferable to Boolean selection if the input data for the selection criterion is unreliable.

Quorum evaluation incidentally illustrates the necessity of searching; it is a calculation which cannot in general be supported by file indexes.

DCI helps the user to infer some facts from a mass of detailed data. CAFS' ability to accumulate a number of counts can be used to profile the values of a particular data item, to demonstrate the shape of the data. A distribution function can be built up by one or more CAFS scans; the results can then be displayed, say via Querymaster, in histogram, pie-chart or some other preferred form of management graphics. A subsequent interaction might focus on some sector of the distribution in search of greater detail; users might include market researchers, quality-control engineers and statisticians.

Turning now to the software industry, a number of software houses are actively working to interface their packages to CAFS via DCI. It supports fully dynamic data-management tools and has low runtime overheads. Logica with Rapport are the first to bring a CAFS version of their product to market; others are in the pipeline.

Where packages have not been developed to exploit DCI implicitly, the user may still have the opportunity to do so if the product is an 'open' one with user procedures. The QuickBuild components Application Master, Reportmaster and Querymaster all come in this category; they can call DCI modules direct as well as using RCI via Cobol. The two CAFS interfaces QM and DCI can be usefully combined together to give both a relational interface and full CAFS selection capability.

### *4.3 Using Querymaster*

Corbin's paper[16] describes the experience of one large user in setting up and using an end-user service. Although computer literacy is on the increase, the introduction of end-user computing in an organisation may involve a change of culture, attitudes, strategy and procedures.

There is a clear trend to providing higher and higher levels of management and decisionmaking with direct IT support. Data processing (DP) departments are delivering support systems to operational staff, professional IT workers and the middle if not the top ranks of management.

The availability of production and *ad hoc* query facilities is a key component of this system provision. Querymaster supported by CAFS facilitates *in situ* enquiry on core corporate data, data extraction from core data to information centres and enquiry within an information centre.

The Querymaster documentation clearly defines the 'queryview controller' role of the individual responsible for setting up one or more end-user query services. His use of such queryview-tailoring facilities as subsetting, renaming and macros can help end-users significantly.

Experience at Southern Water[16] shows its staff adding use of the query service to their regular work pattern. Where query requirements can be covered by a number of predefined macros, the site might consider integrating that query activity with existing TP services through use of RCI.

DP departments will find that good core systems attract peripheral data use of an occasional or *ad hoc* nature. The profile of data use may even change. They will also find that the provision of ready data access highlights the quality of the existing data or the lack of it; the importance of this most intangible of corporate resources will be very obvious.

### *4.4 Using the relational CAFS interface*

RCI presents serial files to a Cobol program in the same way as Querymaster presents tables of information to the end user.

RCI can therefore be used simply to boost the performance of programs which are already searching in serial fashion. This is similar in style to CSO's use except that a simple transformation of the code is required, substituting RCI calls for the existing Cobol calls.

RCI can provide single-stream input from multiple files as required by such products as Reportmaster (RM). It can select on text at word level without the Cobol program needing to recognise this new format. It can simplify the read-only manipulation of IDMS data, substituting a serial-file interface for Codasyl DML (data-manipulation language) and its implicit and sometimes difficult currency concept.

The Viewdata interface is perhaps the most attractive interface to computer systems for the general public; it has a directive screen format in familiar colour television packaging. Viewdata applications on ICL mainframes use the Bulletin TP/Cobol application product.

Berkshire County Council were the first to couple Viewdata and CAFS' search power using RCI; they have provided the public with a CAFS-based library catalogue searchable from some ten terminals in Reading's new central library, replacing constrained searching by author, subject or title by free matching on title. Berkshire demonstrated the value of the Querymaster/RCI duality; they prototyped the program's relational interface with QM and successfully ran the system three working days after taking receipt of RCI.

The deliberate similarities between QM and RCI have another benefit. As the pattern of *ad hoc* and production enquiry identifies itself, a QM service can be partially replaced by a screen-based production query service using RCI. This has usability and operational benefits in that form-filling is simpler than the QM language syntax and the TP environment is more closely controlled than the MAC environment.

RCI has raised the level of the interface between program and data, with three distinct effects. It has enabled programmers to get their programs right earlier; it has removed sources of error, replacing program-coding by a DDS-definition process, and it has increased the independence between program and data, insulating code from such changes as a migration from files to database or vice versa. It has also brought discernible productivity improvements by taking on the common chores of data manipulation which have to date been the lot of all programmers.

A major and skilled site, estimating a telephone-directory enquiry system as a 2 man-year project, completed the work in 13 man-days by combining fourth-generation QuickBuild MMI techniques with RCI/CAFS.

### *4.5 System synthesis*

A wide range of design considerations have been described elsewhere[6,18,21,22]. This section therefore confines itself to the framework for information system analysis and design.

A suitable framework will include a methodology and will support the use of a range of development tools. The development process will identify a number of phases, for example:

> IT strategy, system prioritisation, business system specification, business requirements specification, design, implementation, testing, transition, live running, maintenance.

Each phase will have defined input, output, decision points and criteria against which to optimise; most importantly, there will be criteria defining whether to proceed, double back or abandon the development process.

Given a clear framework, it should be clear where CAFS-related decisions

need to be made. Although different methodologies differ even in their naming and bounding of the above phases, some comments can be made. Note particularly that it is always possible to lose the clarity of the phased approach by making low-level decisions too early.

In the CAFS context, the business-system specification giving the high-level definition of the system's scope can be more ambitious. A broad range of medium-priority requirements will be supported without major design effort. In the business-requirements specification, CAFS will improve the feasible performance targets that can be defined. Finally, CAFS simplifies the design process by removing the need for some of the data infrastructure with consequent benefits for the remaining phases.

## 5 Future directions

The CAFS system can be developed in many ways. This section is the author's personal view of the options available.

On the hardware front, it is reasonable to assume that future implementations of the current CAFS engine will keep pace with disc and data-input technology, taking advantage of VLSI and more powerful constituent microprocessors. As the balance of system costs changes, it is possible that CAFS capability could migrate further from the system centre, from the disc control module to the disc drive itself.

CAFS microcode improvements have already come through in the product.

### *5.1 Further integration*

A likely feature of ICL's product development will be the further integration between the components of the CAFS system and between the CAFS system and its environment. Already, VME has extended the range of CAFS-searchable files and improved coexistence between CAFS and TP activity; both IDMS and Querymaster have been further tuned to facilitate CAFS searches. ICL's intention is that DCI will be supported by DDS and extended to IDMS data; in the wake of DCI, the software industry is integrating a number of packages with CAFS.

The further integration of indexing and CAFS-searching techniques is an interesting prospect, covered in detail elsewhere[21,23,24]. A 'secondary index' or 'coarse index', a search cell index rather than a record index, focuses CAFS searching onto relevant subsections of a file. This technique has already been essential on major projects with large files[24] and could usefully be made generally available as a development of VME's RECMAN facilities. Tagg[21] models the performance impact of secondary indexing.

Knowledge-engineering technology will have an increasing role to play in the future CAFS system. Pilot systems have been developed using Adviser for

performance sizing and end-user guidance. In the latter case, a dialogue with the user leads to the generation of the appropriate Querymaster command.

### *5.2 Text and office*

The CAFS engine was designed with both data and text in mind. Querymaster and RCI handle elementary text on a small scale while DCI provides more comprehensive text-manipulation facilities. CAFS-exploiting 'text' tools and applications have not yet surfaced, although Kay[23] gives an indication of the possibilities.

CAFS development will continue in the context of future international standards on character-representation and text management.

### *5.3 Relational database engine*

The current CAFS engine performs selection and projection on physical records or relational tables. Babb[25] has shown how further specialised hardware can be developed to perform these operations on logical records or general relational views. The hardware simulates joining data by performing selection on a virtual join.

International standards are crystallising at the Structured Query Language (SQL) level; SQL is likely to be more useful as a meeting point for the computer industry than as a language for computer users. The definition of standards in this and other areas encourages the development of customised hardware systems to support those interfaces.

## 6 Summary and conclusions

This paper has defined ICL's current CAFS system, has described the functionality of its hardware and software components and has shown how they work together and in the context of the VME environment. It has given some indication of the way the software interfaces are exploited, as a prelude to the other articles in this volume.

CAFS successfully performs the generic process of data searching with data-driven parallel hardware. Encouraged by the story so far, the author has hazarded an opinion of CAFS future development.

We live in an age of increasing hardware design capability. With increased customisation, we may look forward to the classic general-purpose von Neumann mainframe being replaced by open systems of specialised hardware-based servers. The CAFS system is an example of such a server.

## Acknowledgment

In my turn, I acknowledge with pleasure the significant efforts of past and

present colleagues whose combined skills have produced the CAFS system. I thank Tom Lake, now of Intercept Systems, for his first-hand insights into the architecture of ICL's relational software and the ICLCUA(UK) CAFS Working Party for stimulating discussions over the past four years. Finally, I thank Jack Howlett for his comments on this paper but claim sole credit for any remaining errors and omissions.

## References

1 CARMICHAEL, J.W.S.: 'History of the ICL content-addressable file store (CAFS)', *ICL Tech. J.*, 1985, **4** (4), 352–357.
2 SCARROTT, G.G.: 'Wind of change', *ICL Tech. J.*, 1978, **1** (1), 35–49.
3 MALLER, V.A.J.: 'The content addressable file store – CAFS', *ICL Tech. J.*, 1979, **1** (3), 265–279.
4 CARMICHAEL, J.W.S.: 'Personnel on CAFS: a case study', *ICL Tech. J.*, 1981, **2** (3), 244–252.
5 CROCKFORD, L.E.: 'Associative data management system', *ICL Tech. J.*, 1982, **3** (1), 82–96.
6 ICL Computer Users Association (UK) CAFS SIG: 'Exploiting CAFS-ISP', Working Party Report, July 1984 (2nd Amended Reprint, July 1985), ICLCUA (UK), PO Box 42, Bracknell, Berks. RG12 2LQ, UK.
7 'VME programmer's guide', ICL Technical Publication R00475/01, 1985.
8 'Direct CAFS Interface programming guide (DCI.100)', ICL Technical Publication R00421/01, 1985.
9 'Direct CAFS Interface reference card (DCI.100)', ICL Technical Publication R00431/01, 1985.
10 HUTT, A.T.F.: 'History of the CAFS relational software', *ICL Tech. J.*, 1985, **4** (4), 358–364.
11 CODD, E.F.: 'Relational database: a practical foundation for productivity' (1981 ACM Turing Award Lecture), *CACM*, 1982, **25** (2), Feb.
12 CODD, E.F.: 'Extending the database relational model to capture more meaning', *ACM Trans. Database Syst.*, 1979, **4** (4), 397–434.
13 BABB, E.: 'Joined normal form: a storage encoding for relational databases', *ACM Trans. Database Syst.*, 1982, **7** (4), 588–614.
14 'Using DDS to prepare a query view (QM.250)', ICL Technical Publication R00434/01, 1985.
15 'The Relational CAFS Interface: user guide (RCI.100)', ICL Technical Publication R00251/01, 1985.
16 CORBIN, C.E.H.: 'Creating an end-user CAFS service', *ICL Tech. J.*, 1985, **4** (4), 441–454.
17 'Data Dictionary System: the DDS model (DDS.700)', ICL Technical Publication R00408/01, 1985.
18 ICL Computer Users Association (UK) CAFS SIG: 'CAFS in action', Nov 1985, ICLCUA (UK), PO Box 42, Bracknell, Berks. RG12 2LQ, UK.
19 BURNARD, L.: 'CAFS and text: the view from academia', *ICL Tech. J.*, 1985, **4** (4), 468–482.
20 WALKER, D.: 'Secrets of the sky: the IRAS data at Queen Mary College', *ICL Tech. J.*, 1985, **4** (4), 483–488.
21 TAGG, R.M.: 'CAFS-ISP: issues for the application designer', *ICL Tech. J.*, 1985, **4** (4), 402–418.
22 'CAFS exploitation – a practical guide', ICL Technical Publication R30053/01, 1985.
23 KAY, M.H.: 'Textmaster – a document-retrieval system using CAFS-ISP', *ICL Tech. J.*, 1985, **4** (4), 455–467.
24 WILES, P.R.: 'Using secondary indexes for large CAFS databases', *ICL Tech. J.*, 1985, **4** (4), 419–440.
25 BABB, E.: 'CAFS file-correlation unit', *ICL Tech. J.*, 1985, **4** (4), 489–503.

## Bibliography


1 'IDMS part 2: database establishment (IDMS.400/IDMSX.400)', ICL Technical Publication R00154/03 (3rd Amended Reprint), 1985.
2 'IDMS part 3: using a database (IDMS.400/IDMSX.400)', ICL Technical Publication R00155/03 (3rd Amended Reprint), 1985.
3 'IDMS part 4: database programming (IDMS.400/IDMSX.400)', ICL Technical Publication R00156/03 (3rd Amended Reprint), 1985.
4 'IDMS part 5: database design (IDMS.400/IDMSX.400)', ICL Technical Publication R00153/03 (3rd Amended Reprint), 1985.
5 'Using Querymaster (QM.250)', ICL Technical Publication R00433/01, 1985.
6 'Running Querymaster in VME (QM.250)', ICL Technical Publication R00435/01, 1985.
7 'Querymaster (QM.250) user's reference card', ICL Technical Publication R00436/01, 1985.


## Appendix 1

*Connectivity, coexistence and performance*

The trend has been to increase mainframe to disc channel connectivity. This increases the accessibility of the data and reduces the performance-interference between separate processes. Greater mainframe-CAFS connectivity also implies that a major search task can be syndicated to a larger battery of CAFS engines working in concert; ten engines can search about 25 Mbytes/s.

The CAFS engine attaches to the DCM on 2966-family machines (2953, 2957, 2958, 2966 and 2988) and on Series 39. Single-OCP 2900s can connect to six CAFS engines; dual and superdual configurations can connect to eight engines. On Series 39, the connectivity is greater. Single-node Level 30s can connect to 36 CAFS engines and Level 80s to 72.

The Series 39 DCM is the high-speed disc controller (HSDC) and connects to one CAFS engine. On the 2900, the DCU/2 disc-control unit and more commonly the decision support controller (DSC) unit connect to two CAFS-compatable DCMs.

The maximum disc-drive strings on the various controllers are:

- HSDC: 8*FDS 300 or 4*FDS 2500 or 16 MDSS 'retained' drives
  MDSS drives are EDS 80s, FDS 160s or FDS 640s
- DCU/2 DCM: 16 MDSS drives
- DSC DCM: 32 MDSS drives

We have already seen that some of the synergy and coexistence objectives are being met. CAFS attaches to standard DCMs working with standard discs. It is also the case that CAFS searches most standard RECMAN files, PDS databases and IDMS databases. In the case of IDMS, a single-pass reformat procedure sets up areas of the database for CAFS searching.

Coexistence objectives also require that a 'long' CAFS search should be

interruptible by a 'short' transaction processing task. It would probably be undesirable for a single-record fetch to queue behind a 40-track full-cylinder CAFS scan on an FDS 640. VME therefore fragments CAFS searches, each fragment searching consecutive blocks on a disc-cylinder and being no longer than a system-parameter defined number of tracks.

On 2966s etc., the maximum-fragment parameter default of ten tracks can be changed at system setup time to any value. The value must be chosen to balance the needs of the existing workload against the need to exploit CAFS searching.

On Series 39, multiblock fetches and CAFS searches travel second class; single block fetches travel first class. The HSDC exercises the right to pre-empt long tasks on the disc channel when a short task arrives. Given this degree of HSDC intelligence, the maximum-fragment parameter is unnecessary.

CAFS can search data at some 3·6 Mbytes/s, outrunning the delivery rate of the fastest FDS 2500 drives. In practice, therefore, the following parameters always affect the data search speed of a CAFS engine:

- DS: the maximum formatted-data delivery rate of the disc drive
- BF: the blocking factor; block-size choice effect on delivery rate
- FF: the fragment factor; governed by the temporal dissection of the CAFS search into search fragments
- PF: packing factor; the proportion of the data blocks occupied by the records relevant to the CAFS search.

The basic upper limit on searching speed is therefore:

$$\text{file search rate} \leqslant \text{DS*BF*FF Mbytes/s}$$
$$\text{data search rate} \leqslant \text{DS*BF*FF*PF Mbytes/s}$$

Other factors such as disc head movement, rotational latency, file fragmentation, buffer management and process multiplexing all subtract from the data rate as perceived by the application program or the end user. Note, however, that these aspects of performances are standard and preceded the introduction of CAFS.

The disc speeds DS of the drives are 1·17696 Mbytes/s (MDSS), 2·22910 Mbytes/s (FDS 300) and 2·83277 Mbytes/s (FDS 2500).

The blocking factor BF is also specific to the drive concerned. Below are listed BFs for the three drives where the block size is chosen as $N$ kbytes or as a maximal value for that number of blocks/track:

Blocking factors are more significant on the faster drives and the best block sizes vary from drive to drive.

**MDSS disc drives**

| Block size, bytes | Blocks/ track | Bytes/ track | Blocking factor |
|---|---|---|---|
| 2048 | 9 | 18 432 | 0·939641 |
| 3072 | 6 | 18 432 | 0·939641 |
| 4096 | 4 | 16 384 | 0·835237 |
| 6144 | 3 | 18 432 | 0·939641 |
| 9216 | 2 | 18 432 | 0·939641* |
| 18 432 | 1 | 18 432 | 0·939641 |
| 2057 | 9 | 18 513 | 0·943770 |
| 3155 | 6 | 18 930 | 0·965029 |
| 4801 | 4 | 19 204 | 0·978997 |
| 6447 | 3 | 19 341 | 0·985981 |
| 9739 | 2 | 19 478 | 0·992965 |
| 19 616 | 1 | 19 616 | 1·000000 |

**FDS 300 disc drives**

| Block size, bytes | Blocks/ track | Bytes/ track | Blocking factor |
|---|---|---|---|
| 2048 | 15 | 30 720 | 0·804525 |
| 3072 | 10 | 30 720 | 0·804525 |
| 4096 | 8 | 32 768 | 0·858160 |
| 6144 | 5 | 30 720 | 0·804525 |
| 9216 | 4 | 36 864 | 0·965431* |
| 18 432 | 2 | 36 864 | 0·965431 |
| 2100 | 15 | 31 500 | 0·824953 |
| 3412 | 10 | 34 120 | 0·893568 |
| 4404 | 8 | 35 232 | 0·922690 |
| 7316 | 5 | 36 580 | 0·957993 |
| 9300 | 4 | 37 200 | 0·974230 |
| 19 092 | 2 | 38 184 | 1·000000 |

**FDS 2500 disc drives**

| Block size, bytes, | Blocks/ track | Bytes/ track | Blocking factor |
|---|---|---|---|
| 2048 | 18 | 36 864 | 0·785142 |
| 3072 | 13 | 39 936 | 0·850570 |
| 4096 | 10 | 40 960 | 0·872380 |
| 6144 | 7 | 43 008 | 0·915999* |
| 9216 | 4 | 36 864 | 0·785142 |
| 18 432 | 2 | 36 864 | 0·785142 |
| 2164 | 18 | 38 952 | 0·829613 |
| 3188 | 13 | 41 444 | 0·882688 |
| 4276 | 10 | 42 760 | 0·910717 |
| 6356 | 7 | 44 492 | 0·947606 |
| 11 476 | 4 | 45 904 | 0·977679 |
| 23 476 | 2 | 46 952 | 1·000000 |

The fragment factor (FF) reflects the fact that the CAFS search is fragmented by VME. The interfragment overhead is typically two disc revolutions or some 33 ms:

$$FF = (\text{max fragment size})/(\text{max fragment size} + 2)$$

Taking into account the parameters DS, BF and FF and adopting the asterisked block sizes, we can calculate 'typical' effective CAFS file search speeds as 0·922 Mbytes/s (MDSS), 1·793 Mbytes/s (FDS 300) and 2·290 Mbytes/s (FDS 2500).

The last parameter to be discussed is the data packing factor PF. Data is not usually 100% packed in a file for many reasons. Varieties of red-tape accompany the object data, several record types may coexist in a file and dynamic data should be packed at lower densities to avoid overflow.

Low packing densities have a proportionate effect on the data search rate of CAFS, but this does not mean that they subtract from the value of CAFS. Unused file space and non-target data types are the first examples of CAFS' effectiveness in filtering out disc space which is irrelevant to the search and to any subsequent processing.

The Series 39 figures below indicate retrieval processor times for various tasks. In the worst case of short records and a high hit rate, the RP component of CAFS will not keep pace with the search rate at the CAFS front end.

130 μs = overhead per hit record (260 μs on 2966s etc.)
$(L+8)/3{\cdot}5$ μs = output transfer time for the $L$-byte record
87 μs = overhead per function call
200 μs = function adding totalling on an 8-byte field
250 μs = function comparing two 2-byte fields
500 μs = function comparing two 20-byte fields

When the high hit rate is a local phenomenon, a cluster of records satisfying the LC condition, the buffering within CAFS helps to maintain the output performance of the engine.

**Appendix 2**

*Glossary of abbreviations*

| | |
|---|---|
| ADRAM | alien data record access method |
| AM | Application Master |
| AV | application view (RCI) |
| AVM | application virtual machine |
| BF | blocking factor |
| CAFS | content-addressable file store |
| Codasyl | Conference on data systems languages |

| | |
|---|---|
| CSO | VME CAFS search option |
| DCI | direct CAFS interface |
| DCM | disc control module |
| DCU/2 | disc control unit |
| DDS | data dictionary system |
| DML | data manipulation language (Codasyl, IDMS) |
| DS | disc speed |
| DSC | decision support controller |
| EDS | exchangeable-disc store |
| FDS | fixed-disc store |
| FF | fragment factor |
| HSDC | high-speed disc controller |
| ICLCUA | ICL Computer Users' Association |
| IDMS | integrated data management system |
| IFC | interfield comparisons |
| ISAM | index sequential access method |
| ISP | information search processor |
| KC | key channel |
| LC | logical condition |
| LFU | logical format unit |
| MAMPHY | physical magnetic media |
| MIP | misleading index of performance |
| MMI | man-machine interface |
| OCP | order code processor |
| PDS | personal database system |
| PF | packing factor |
| PLI | programming language instruction |
| QM | Querymaster |
| QP | quorum processor |
| RAM | record access method |
| RCI | relational CAFS interface |
| RECMAN | record manager |
| RM | Reportmaster |
| RP | retrieval processor |
| RSI | restricted system interface |
| RU | retrieval unit |
| RV | relational view (RCI) |
| SCL | system control language |
| SEP | search evaluation processor |
| SEU | search evaluation unit |
| SIF | self-identifying format |
| SP | select processor |
| SQL | Structured Query Language |
| STCC | SET-CAFS-CRITERIA (SCL for CSO) |
| SV | system version (software set) |
| TNF | third normal form |
| TPMS | transaction processing management system |
| VME | virtual machine environment |